

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2001-92796

(P2001-92796A)

(43)公開日 平成13年4月6日(2001.4.6)

(51)Int.Cl. <sup>7</sup>	識別記号	F I	テーマコード*(参考)
G 0 6 F 15/167		G 0 6 F 15/167	B 5 B 0 0 5
12/02	5 1 0	12/02	5 1 0 A 5 B 0 4 5
12/06	5 3 0	12/06	5 3 0 D 5 B 0 6 0
12/08		12/08	H
12/10		12/10	H
審査請求 未請求 請求項の数11 O L (全 23 頁) 最終頁に続く			

(21)出願番号 特願平11-263793

(22)出願日 平成11年9月17日(1999.9.17)

(71)出願人 599074648

ターボデータラボラトリー有限会社

東京都台東区松が谷1-9-12 SPKビルディング604号

(72)発明者 古庄 晋二

神奈川県横浜市神奈川区松見町4丁目1101番地7コートハウス菊名804号

(74)代理人 100103632

弁理士 窪田 英一郎 (外1名)

Fターム(参考) 5B005 KK14 KK22 MM32

5B045 BB12 GG14

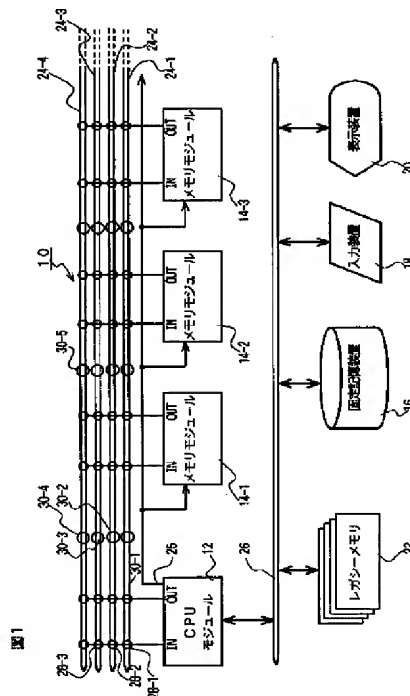
5B060 MB04

(54)【発明の名称】 並列コンピュータのアーキテクチャおよびこのアーキテクチャを利用した情報処理ユニット

(57)【要約】

【課題】 分散メモリ型において、著しく高速な並列処理を実現可能なコンピュータアーキテクチャを提供する

【解決手段】 コンピュータシステム10は、CPUモジュール12と、それぞれがMPU36およびRAMコア34とを有する複数のメモリモジュール14と、CPUとメモリモジュールとの接続やメモリモジュール間の接続をなす複数組のバス24とを備え、CPU12から与えられるインストラクションにより、各メモリモジュールが作動する。所定の関連を有する一連のデータには、空間IDが付与され、各メモリモジュールが、少なくとも、当該空間ID、自己が管理する一連のデータの部分に関する論理アドレス、一連のデータのサイズを含むテーブルを管理し、かつ、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、RAMコアに記憶されたデータに関する処理を実行する。



## 【特許請求の範囲】

【請求項1】 CPUモジュールと、それぞれがMPUおよびRAMコアとを有する複数のメモリモジュールと、前記CPUとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、CPUから各メモリモジュールのMPUに与えられるインストラクションにより、各メモリモジュールのMPUが作動するように構成された並列コンピュータのアーキテクチャであって、

所定の関連を有する一連のデータに、空間IDが付与され、各メモリモジュールのMPUが、少なくとも、当該空間ID、自己が管理する一連のデータの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、各メモリモジュールのMPUが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、RAMコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをRAMコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする並列コンピュータのアーキテクチャ。

【請求項2】 前記MPUが、CPUから与えられた空間IDを、自己が管理する1以上の一連のデータの空間IDと比較する空間コンパレータと、CPUから与えられた論理アドレスと、自己が管理するデータの部分の論理アドレスとを比較するアドレスコンパレータと、当該論理アドレスに基づき、自己のRAMセル上の物理アドレスを算出するアドレスカリキュレータとを有することを特徴とする請求項1に記載のコンピュータアーキテクチャ。

【請求項3】 前記メモリモジュールの各々が、CPUモジュールおよび他のメモリモジュールとの同期をなすための同期信号を受け入れ、かつ、前記複数組のバスの何れかとの接続が可能な入力と、前記複数組のバスの他の何れかとの接続が可能な出力を備え、少なくとも、前記同期信号にしたがって、前記何れかのバスと入力との接続により、データを入力しつつ、前記他の何れかのバスと出力との接続により、データを出力できるように構成されたことを特徴とする請求項1または2に記載のコンピュータアーキテクチャ。

【請求項4】 前記複数組のバスの各々に、前記CPUモジュールと何れかのメモリモジュールの入力または出力との間、および／または、他の何れかのメモリモジュールの入力または出力と、さらに他のメモリモジュールの出力または入力との間の接続を規定するためのスイッチが設けられ、

前記スイッチの切換により、複数組のバスの各々において、並列的にデータの授受が実現されることを特徴とす

る請求項3に記載のコンピュータアーキテクチャ。

【請求項5】 前記複数組のバスのうちの何れかである第1のバスに、何れかのメモリモジュールの出力と、他の何れかのメモリモジュールの入力とが接続され、かつ、前記複数組のバスのうち、他の何れかである第2のバスに、当該他の何れかのメモリモジュールの出力と、さらに他の何れかのメモリモジュールの入力とが接続され、第1のバスにおけるデータの授受と、第2のバスにおけるデータの授受が並列的に進行することを特徴とする請求項4に記載のコンピュータアーキテクチャ。

【請求項6】 前記バスとメモリモジュールとの間の接続を繰り返して、多段のメモリモジュール間の接続を形成することを特徴とする請求項5に記載のコンピュータアーキテクチャ。

【請求項7】 前記MPUが、一連のデータ中の特定の要素を削除し、前記一連のデータ中に特定の要素を挿入し、或いは、一連のデータの末尾に特定の要素を追加することを示すインストラクションを受理すると、テーブルを参照して、自己の管理するデータの領域と、削除、挿入或いは追加にかかる要素の位置とを比較して、当該比較結果に応じて、前記テーブルの内容を更新することを特徴とする請求項1ないし6の何れか一項に記載のコンピュータアーキテクチャ。

【請求項8】 前記MPUが、与えられたインストラクションに回答して、一連のデータ中の要素を特定するための添え字を変換し、および／または、要素に特定の修飾を与える値変換を実行することを特徴とする請求項1ないし7の何れか一項に記載のコンピュータアーキテクチャ。

【請求項9】 CPUモジュールと、それぞれがMPUおよびRAMコアとを有する複数のメモリモジュールと、前記CPUとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、CPUから各メモリモジュールのMPUに与えられるインストラクションにより、各メモリモジュールのMPUが作動するように構成された情報処理ユニットであって、

所定の関連を有する一連のデータに、空間IDが付与され、各メモリモジュールのMPUが、少なくとも、当該空間ID、自己が管理する一連のデータの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、各メモリモジュールのMPUが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、RAMコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをRAMコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする情報処理ユニット。

【請求項10】 前記CPUモジュールが、レガシーメ

メモリ、入力装置および表示装置を相互接続する他のバスと連結可能に構成されたことを特徴とする請求項9に記載の情報処理ユニット。

【請求項11】 請求項9に記載の情報処理ユニットと、CPUモジュールと他のバスを介して連結された1以上のレガシーメモリを含む記憶装置、入力装置および表示装置とを有することを特徴とするコンピュータシステム。

【発明の詳細な説明】

【0001】

【産業上の技術分野】本発明は、SIMD(Single Instruction Stream, Multiple Data Stream)を実現可能な並列コンピュータのアーキテクチャに関し、より詳細には、適切かつ高速なメモリ制御により、汎用的な並列演算が可能なコンピュータアーキテクチャに関する。

【0002】

【従来の技術】社会全体のさまざまな場所にコンピュータが導入され、インターネットをはじめとするネットワークが浸透した今日では、そこそこで、大規模なデータが蓄積されるようになった。このような大規模データを処理するには、膨大な計算が必要で、そのために並列処理を導入しようと試みるのは自然である。

【0003】さて、並列処理アーキテクチャは「共有メモリ型」と「分散メモリ型」に大別される。前者(「共有メモリ型」)は、複数のプロセッサが1つの巨大なメモリ空間を共有する方式である。この方式では、プロセッサ群と共有メモリ間のトラフィックがボトルネックとなるので、百を越えるプロセッサを用いて現実的なシステムを構築することは容易ではない。したがって、例えば10億個の浮動小数点変数の平方根を計算する際、単一CPUに対する加速比は、せいぜい100倍ということになる。経験的には、30倍程度が上限である。後者(「分散メモリ型」)は、各プロセッサがそれぞれローカルなメモリを持ち、これらを結合してシステムを構築する。この方式では、数百～数万ものプロセッサを組み込んだハードウェアシステムの設計が可能である。したがって、上記10億個の浮動小数点変数の平方根を計算する際の単一CPUに対する加速比を、数百～数万倍とすることが可能である。しかしながら、後者においても、後述するいくつかの課題が存在する。本出願は、「分散メモリ型」に関するものであり、この方式について最初に多少の考察を加えながら従来技術との比較を行うことにする。

【0004】

【課題を解決するための手段】[第1の課題：巨大配列の分掌管理]「分散メモリ型」の第1の課題は、データの分掌管理の問題である。巨大なデータ(一般的には配列なので、以降、配列で説明する)は、1つのプロセッサの所有するローカルメモリに収容できるものではなく、必然的に複数のローカルメモリに分掌管理される。

効率的かつ柔軟な分掌管理メカニズムを導入しないと、プログラムの開発および実行に際してさまざまな障害を抱え込むことになることは明らかである。

【0005】[第2の課題：プロセッサ間通信の効率の低さ]分散メモリ型システムの各プロセッサが、巨大配列にアクセスしようとする、自己の所有するローカルメモリ上の配列要素に対しては速やかにアクセスできるものの、他のプロセッサが所有する配列要素へのアクセスはプロセッサ間通信を必須とする。このプロセッサ間通信はローカルメモリとの通信に比べ、極端にパフォーマンスが低く、最低でも100クロックかかると言われる。このため、ソート実施時には、巨大配列全域にわたる参照が実施され、プロセッサ間通信が多発するため、パフォーマンスが極端に低下する。

【0006】この問題点につき、より具体的に説明を加える。1999年現在、パソコンは、1～数個のCPUを用いて、「共有メモリ型」として構成されている。このパソコンに使用される標準的なCPUは、メモリバスの5～6倍程度の内部クロックで動作し、その内部に自動的な並列実行機能やパイプライン処理機能が装備されており、およそ1データを1クロック(メモリバス)で処理できる。「共有メモリ型」であるパソコンにて巨大配列のソート処理を行う場合、1データについて1クロックを要し、このため、1データに100クロック(メモリバス)を要する。「分散メモリ型」のマルチプロセッサシステムの100倍のパフォーマンスを発揮することも考えられる。

【0007】[第3の課題：プログラムの供給]「分散メモリ型」の第3の課題は、多数のプロセッサにどうやってプログラムを供給するか、という問題である。非常に多数のプロセッサに、別々のプログラムをロードし、全体を協調動作させる方式(MIMD: Multiple Instruction Stream, Multiple Data Stream)では、プログラムの作成、コンパイル、配信のために多大な負荷を要する。その一方、多数のプロセッサを同一のプログラムで動作させる方式(SIMD: Single Instruction Stream, Multiple Data Stream)では、プログラムの自由度が減少し、所望の結果をもたらすプログラムが開発できない事態も想定される。

【0008】本発明は、「分散メモリ型」の上記第1ないし3の課題を解決する方法およびコンピュータアーキテクチャを提供する。第1の「巨大配列の分掌管理」の課題は、配列の各要素の配置(物理アドレス)を、各プロセッサモジュールが統一的方法で分掌管理することで解決できる。この手法により、ガーベジコレクションの必要性が無くなり、配列要素の挿入・削除が数クロックで完了し、SIMDを実現する上で欠かせない各プロセッサの暗黙の(非明示的)処理分担を割り付けることもできる。この方法は、後ほど「多空間メモリ」という概念で説明される。

【0009】第2の「プロセッサ間通信の効率の低さ」の課題は、達成しようとする処理に応じて各プロセッサ間をつなぎ替え、各接続経路毎に、定められた種類のデータを、定められた順番で、1方向に連続転送することで、バスの能力を100%近くまで使用できるよう通信をスケジュール化し、同時に巨大パイプライン処理を実現することで解決できる。その有効性を実証するため、後ほど、現実的なシステム設計で、10億行のソートを1秒程度で完了するシステムの構成方法を例示するであろう。これは、既知の最高速の装置に比べて、1万倍以上高速である。この方法は、後ほど「組替えバス」技術として説明される。

【0010】第3の「プログラムの供給」の課題は、SIMD方式を採用することで解決できる。SIMDの場合は、各プロセッサの暗黙の（非明示的）処理分担をどうやって決定するか？が最大の問題であるが、前述の「多空間メモリ」技術にてこの処理分担が自動的に決定でき、SIMDであってもプログラムの自由度を保持することができる。つまり、本発明は、分散メモリ型において、単一命令により種々のメモリに記憶された配列中の要素を入出力し、著しく高速な並列処理を実現可能なコンピュータアーキテクチャを提供することを目的とする。

【0011】

【課題を解決するための手段】本発明の目的は、CPUモジュールと、それぞれがMPUおよびRAMコアとを有する複数のメモリモジュールと、前記CPUとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、CPUから各メモリモジュールのMPUに与えられるインストラクションにより、各メモリモジュールのMPUが作動するように構成された並列コンピュータのアーキテクチャであって、所定の関連を有する一連のデータに、空間IDが付与され、各メモリモジュールのMPUが、少なくとも、当該空間ID、自己が管理する一連のデータの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、各メモリモジュールのMPUが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、RAMコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをRAMコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする並列コンピュータのアーキテクチャにより達成される。

【0012】本発明によれば、空間IDを用いて一連のデータを把握するため、当該一連のデータが、多数のメモリモジュールにより分掌されても、各メモリモジュールのMPUが、当該一連のデータを確実に認識することができる。また、メモリモジュールは、一連のデータお

よび自己が管理するその部分を、テーブルにて把握しているため、インストラクションの受理にしたがって、そのテーブルを参照して、所定の処理を実行することができる。これにより、単一インストラクションに基づく、各MPUでの並列処理が実現できる。

【0013】本発明の好ましい実施態様においては、MPUは、CPUから与えられた空間IDを、自己が管理する1以上の一連のデータの空間IDと比較する空間コンパレータと、CPUから与えられた論理アドレスと、自己が管理するデータの部分の論理アドレスとを比較するアドレスコンパレータと、当該論理アドレスに基づき、自己のRAMセル上の物理アドレスを算出するアドレスカリキュレータとを有している。これらコンパレータおよびカリキュレータは、ハードウェアにて構成されても良いし、MPUのプログラムによりソフトウェアとして実現されるものであっても良い。

【0014】また、本発明の好ましい実施態様においては、メモリモジュールの各々が、CPUモジュールおよび他のメモリモジュールとの同期をなすための同期信号を受け入れ、かつ、前記複数組のバスの何れかとの接続が可能な入力と、前記複数組のバスの他の何れかとの接続が可能な出力を備え、少なくとも、前記同期信号にしたがって、前記何れかのバスと入力との接続により、データを入力しつつ、前記他の何れかのバスと出力との接続により、データを出力できるように構成されている。本実施の形態によれば、同期信号にしたがって、メモリモジュールからのデータ出力およびメモリモジュールへのデータ入力となされ、かつ、バスの接続の制御により、適切に並列処理を実現することが可能となる。

【0015】複数組のバスの各々には、前記CPUモジュールと何れかのメモリモジュールの入力または出力との間、および／または、他の何れかのメモリモジュールの入力または出力と、さらに他のメモリモジュールの出力または入力との間の接続を規定するためのスイッチが設けられ、スイッチの切換により、複数組のバスの各々において、並列的にデータの授受が実現されるのがより好ましい。これにより、複数組のバスをより有効に利用することが可能となり、より並列性を高めることが可能となる。

【0016】本発明のさらに好ましい実施態様においては、複数組のバスのうちの何れかである第1のバスに、何れかのメモリモジュールの出力と、他の何れかのメモリモジュールの入力が接続され、かつ、前記複数組のバスのうち、他の何れかである第2のバスに、当該他の何れかのメモリモジュールの出力と、さらに他の何れかのメモリモジュールの入力が接続され、第1のバスにおけるデータの授受と、第2のバスにおけるデータの授受が並列的に進行する。このように、コンピュータの実施態様によれば、CPUモジュールと、メモリモジュールとにより、パイプライン処理を実現することが可能と

10

20

30

40

50

なる。バスとメモリモジュールとの間の接続を繰り返して、多段のメモリモジュール間の接続を形成するのがより好ましい。

【0017】本発明の別の好ましい実施態様においては、MPUが、一連のデータ中の特定の要素を削除し、前記一連のデータ中に特定の要素を挿入し、或いは、一連のデータの末尾に特定の要素を追加することを示すインストラクションを受理すると、テーブルを参照して、自己の管理するデータの領域と、削除、挿入或いは追加にかかる要素の位置とを比較して、当該比較結果に応じて、前記テーブルの内容を更新する。すなわち、MPUにおいて、自己が管理するテーブルを更新する、すなわち、リマッピングをすることにより、要素の削除、挿入および追加を実現することが可能となる。

【0018】本発明のさらに別の実施態様においては、MPUが、与えられたインストラクションに応答して、一連のデータ中の要素を特定するための添え字を変換し、および／または、要素に特定の修飾を与える値変換を実行する。また、本発明の目的は、CPUモジュールと、それぞれがMPUおよびRAMコアとを有する複数のメモリモジュールと、前記CPUとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、CPUから各メモリモジュールのMPUに与えられるインストラクションにより、各メモリモジュールのMPUが作動するように構成された情報処理ユニットであって、所定の関連を有する一連のデータに、空間IDが付与され、各メモリモジュールのMPUが、少なくとも、当該空間ID、自己が管理する一連のデータの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、各メモリモジュールのMPUが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、RAMコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをRAMコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする情報処理ユニットによっても達成される。たとえば、前記ユニットが単一の回路基板に形成され、CPUモジュールが、レガシーメモリ、入力装置および表示装置を相互接続する他のバスと連結可能に構成されていても良い。

【0019】さらに、本発明の目的は、上記情報処理ユニットと、CPUモジュールと他のバスを介して連結された1以上のレガシーメモリを含む記憶装置、入力装置および表示装置とを有することを特徴とするコンピュータシステムによっても達成される。

【0020】

【発明の実施の形態】〔ハードウェア構成〕以下、添付図面を参照して、本発明の実施の形態につき説明を加え

る。図1は、本発明の実施の形態にかかるコンピュータシステムの構成を示すブロックダイアグラムである。図1に示すように、コンピュータシステム10は、単一命令による並列演算を実現するCPUモジュール12と、並列演算のために必要な種々のデータを記憶するメモリモジュール14-1、14-2、14-3、…と、必要なプログラムやデータを記憶する固定記憶装置16と、キーボードやマウスなどの入力装置18と、CRTなどからなる表示装置20と、種々の形式のデータ等が記憶されているレガシーメモリ22とを備えている。また、バス24-1、24-2、…において、CPUモジュール12、各メモリモジュール14との接点には、スイッチ28-1、28-2、28-3、…などが配設され、選択された回路要素間における情報の授受が可能となっている。また、CPUモジュール12とメモリモジュール14-1との間、隣接するメモリモジュール間において、バスの連結および接続をなすためのスイッチ30-1、30-2、…が設けられている。

【0021】CPUモジュール12と、メモリモジュール14との間には、複数のバス24-1、24-2、24-3、24-4、…とが設けられている。したがって、CPUモジュール12とメモリモジュール14との間、および、メモリモジュール間は、上記バスによりデータ等の授受が可能となっている。また、CPU12と、メモリモジュール14との間には、制御信号ライン25が設けられ、CPU12から発せられるインストラクションなどが、全てのメモリモジュール14に伝達されるようになっている。

【0022】さらに、CPU12と、他の構成要素（たとえば、固定記憶装置16、入力装置18など）との間には、ローカルバス26が配設されており、これらの間でもデータ等の授受が可能となっている。CPU12は、固定記憶装置16に記憶され、或いは、バス26上に接続されたRAMのような他の記憶装置（図示せず）に記憶されたプログラムを読み出し、このプログラムにしたがって、以下に示すメモリモジュール14へのインストラクションの送出を含むデータの授受のほか、スイッチ28、30の制御等を実行する。また、CPU12は、プログラムにしたがって、レガシーメモリ22に記憶された種々の形式のデータを受け入れて、この形式のデータを、CPU12、メモリモジュール14、バス24からなる系にて処理可能な一連のデータ（配列）に変換し、これらを、各メモリモジュール14に記憶させることもできる。

【0023】図2は、各メモリモジュール14の概略を示すブロックダイアグラムである。図2に示すように、メモリモジュール14は、CPUモジュール12から与えられるクロックなど同期信号を受け入れるクロックバッファ32と、データを記憶するRAMコア34と、後述する空間IDやデータの要素番号等を把握し、CPU

12からのインストラクションなどを受理した場合に、空間IDや要素番号に基づき、RAMコア34へのデータ書き込みやRAMコアからのデータ読み出しを制御するMPU36と、バスの何れかからのデータを受け入れて、RAMコア34に供給し、および/または、RAMコア34からのデータを何れかのバスに送出するI/O38とを有している。この実施の形態において、メモリモジュール14は、制御信号ライン25を介して、CPUからのインストラクションを受け入れ、MPU36が、このインストラクションに応答して、RAMコア34のデータを読み出し、RAMコア34にデータを書き込み、或いは、データに所定の処理を施すことができるようになっている。また、RAMコア34へのデータアクセスや、I/Oを介してデータ入力およびデータ出力は、クロックバッファ32に与えられるクロックなどの同期信号に基づき実行される。

【0024】図1および図2から明らかなように、本発明において、コンピュータシステム10は、メモリ共有型のシステムであると考えることができる。また、後述するように、制御信号ライン25を介して、各メモリモジュール14にインストラクションを与えることにより、各メモリモジュール14が並列的に処理を実行する。また、バスへのデータ出力およびバスからのデータ入力などが、所定の同期信号に基づき実行される。したがって、このコンピュータシステム10は、SIMDの形態をなしていると考えることができる。

【0025】[実現される機能の概略]このような構成を有するコンピュータシステム10につきより詳細な説明を加える前に、本コンピュータシステム10により実現される機能の概略を簡単に説明する。

#### (1) 多空間メモリ

本明細書において、多空間メモリとは、メモリ空間を、空間IDとアドレスとに基づきアクセスするために割り当てられたメモリ空間をいう。これにより、一連のデータが多数のプロセッサに分掌されていても、各プロセッサが、これを確実に分離、認識することができる。従来のメモリ空間においては、プロセス毎に個別の領域を割り当てることはあっても、一連の変数(配列、構造体など)毎に目盛り空間を割り当てることは行われてこなかった。したがって、以下、このような従来のメモリ空間を「単一メモリ空間」と称する。単一メモリ空間のシステムにおいては、アドレスのみを用いてデータにアクセスしているため、関連を有する一連のデータを分離したり、認識することができなかった。このため、実際には並列処理が可能であっても、その可否を判断できない場合が多かった。また、ある単一メモリ空間に、新たな一連のデータを収容させる場合に、当該一連のデータの収容場所を確保するために、ガーベージコレクションを実行する必要があった。

【0026】これに対して、本発明においては、メモリ

空間に、空間IDを導入し、一連のデータについて同一のIDを付与している。また、メモリモジュール14において、自身のRAMコア34に保持されているデータに関する空間IDを把握し、これにより、各メモリモジュール14自体が、現在アクセスされているデータの空間IDを参照することにより、自己の作動の是非を決定することができる。また、各メモリモジュールが空間IDと関連付けて、一連のデータの全部或いは一部を保持できるため、ある一連のデータを、複数のメモリモジュール14に分割して記憶させることができ、これによりガーベージコレクションを不要にすることができる。

【0027】たとえば、図3に示すように、単一メモリ空間において、“A”という一連のデータ、“B”という一連のデータ、…が収容されている場合を考える。たとえば、ここで、全メモリサイズが32ワードで、上記一連のデータのサイズの総和が30ワードであると仮定する。これら一連のデータは、空間中に点在しているため、未使用のメモリサイズは、12ワードであるにもかかわらず、実際に格納できる一連のデータのサイズは3ワードに限定される。このため、3ワードを超えたサイズを有する新たな一連のデータを収容すべき場合には、ガーベージコレクションを実行しなければならない。その一方、図4に示すように、本発明においては、一連のデータの各々に、空間IDが付与されている。これらは、空間IDと関連付けられて、1以上のメモリモジュール14に記憶される。したがって、未使用のサイズと収容可能なサイズとを一致させることが可能となる。

#### 【0028】(2) メモリモジュール

また、本発明においては、各メモリモジュール14が、MPU36を有し、上記空間IDのほか、自己が保持する一連のデータの各々の要素番号を把握している。したがって、CPU12からのインストラクションを受理した後、MPU36が、インストラクションにしたがってアクセスすべきデータが、自己のRAMコア34中に保持されているものか否かを判断して、アクセスに必要なの是非を決定することができる。さらに、各メモリモジュール14が、自己のRAMコア34に格納されている配列要素の添え字の範囲から、SIMDでのインストラクションにおける暗黙の処理の分担範囲を決定することが可能である。

【0029】また、本発明においては、メモリモジュール14が、アドレスリマッピングを実行できるようになっている。たとえば、図5に示すように、ある配列の所定の位置に特定の要素を挿入する場合、その他、所定の位置の要素を削除し、或いは、配列の末尾に所定の要素を追加する場合にも、本実施の形態においては、当該配列に関連する要素を保持しているメモリモジュールの各々において、MPU36が、アドレスリマッピングを実行することにより、並列的かつ高速に、これらを実現することができる。さらに、図6に示すように、配列の要

素(値)に修飾を与える場合(たとえば、各値に「1」を加える場合)にも、関連する配列の要素を保持するメモリモジュールの各々において、MPU36が、並列的かつ高速に、必要な処理を行うことができる。

【0030】また、メモリモジュール14においては、MPU36が、RAMコア34にて記憶すべきデータの各々のサイズを把握し、圧縮した形態にてこれらを記憶することができる。たとえば、あるメモリモジュール14にて、整数値のデータを保持すべき場合に、実際のデータ値が“0”ないし“3”までの値しか取り得ない場合には、MPU36は、各データのために2ビットのみを用意する。CPU12との間では、1つの整数を表現するために32ビットを使用していた場合には、メモリモジュール14とCPU12の間での通信のために、MPU36が、データ形式を変更して、CPU12との授受をなせば良い。これにより、RAMコア34をより無駄なく利用することが可能となる。また、文字列のような長さの異なるデータについても、同様にデータ長を変更して記憶することができるようになっている。

【0031】さらに、メモリモジュール14においては、所定の空間IDに関連付けられたデータや、所定の範囲の要素番号を付されたデータに、特定の値(たとえば、「0」)をセットすることができるようになっている。これにより、メモリモジュール14内で、高速に初期化の処理を実行することが可能となる。また、メモリモジュール14においては、ある特定のデータ(配列)中の値を検索することや、添字の範囲をチェックすることが可能である。

#### 【0032】(3) 組み替え可能バス

本発明においては、CPU12が、スイッチ28-1、28-2、…およびスイッチ30-1、30-2、…を選択的にオン/オフして、データの授受をなすべきメモリモジュール14を指定することにより、パイプライン処理を実現している。たとえば、図7に示すように、あるメモリモジュール14-iから出力されたデータを、他のメモリモジュール14-jに与え、かつ、当該他のメモリモジュール14-jから出力されたデータを、さらに他のメモリモジュール14-kに伝達すべき場合には、CPU12は、バス24-mを、メモリモジュール14-i、14-jのために割り当て、かつ、バス24-nを、メモリモジュール14-j、14-kのために割り当てるように、各スイッチの状態を設定する。

【0033】さらに、これらパイプライン処理は、単一のメモリモジュール間の接続により実現される場合だけでなく、複数の一連のメモリモジュール(メモリモジュール群)の間の接続により実現することも可能である。達成しようとする処理に応じて、各メモリモジュール間をつなぎ替え、各接続経路毎に、定められた種類のデータを定められた順序にて一方向に連続転送することで、バスの能力を100%近く使用できるように、通信をス

ケジュール化することができる。これにより、分散メモリ型の並列処理システムの最大の問題であった、プロセッサ間通信のパフォーマンスの低さを、解消することができる。このように構成されたコンピュータシステム10において、多空間メモリの具体的構成および多空間メモリにおけるシステムの作動につき説明を加える。

【0034】[多空間メモリ] 図8は、多空間メモリの下での、メモリモジュール14の構造を説明するための図である。図8(a)に示すように、メモリモジュール14中のRAMコア34には、空間ID管理テーブルが設けられる。これにより、メモリモジュール14のMPU36は、自己が保持するデータの空間ID等必要な情報を把握することが可能となる。図8(b)に示すように、空間ID管理テーブルには、自己が保持するデータ群ごとの、空間ID、CPUの管理の下での、データ群の論理開始アドレス、データ群が割り付けられた領域のサイズ、RAMコア34中の物理開始アドレス、当該空間IDを有する一連のデータの全サイズ、および、アクセス制限を示すアクセス制限フラグが格納されている。アクセス制限フラグは、この実施の形態においては、読み出しのみ可能(R)、書き込みのみ可能(R)、読み書き可能(RW)の3つの状態を示すことができるようになっている。

【0035】メモリモジュール14のMPU36は、ある空間IDを有するデータ群が与えられた際に、RAMコア34中に当該データ群を収容すべき、1以上の領域を見出して、当該領域にデータ群をそのまま、或いは、2以上に分割して収容する。この際に、与えられた空間ID、論理開始アドレス、全サイズ、アクセス制限フラグとともに、実際にデータを収容したRAMコア中の論理開始アドレスや、割り付け領域サイズも、空間ID管理テーブルに記憶される。図8(c)は、図8(b)による空間ID管理テーブルにしたがったRAMコア36中のデータを示す図である。

【0036】[メモリアクセスの概略説明] このように構成されたメモリモジュール14へのアクセスにつき以下に説明を加える。図9に示すように、まず、CPU12が、空間IDおよび論理アドレス、並びに、必要なインストラクション(たとえば、データの書き込みや読み出し)を、制御信号ライン25を介して、全てのメモリモジュール14に伝達する。各メモリモジュール14においては、これにตอบสนองして、MPU36に設けられた空間コンパレータ52が、空間IDと、自己の空間ID管理テーブル上に保持されている空間IDとを比較して、同一のものを、自己が保持しているかを判断し、また、アドレスコンパレータ54が、論理アドレスについて、同様の判断を行う。次いで、メモリモジュール14のMPU36が、自己のRAMコア34に、インストラクションによる処理対象となるデータが保持されていると判断した場合には、アドレスカリキュレータ56



が、空間ID管理テーブルを参照して、RAMコア34中の物理アドレスを算出し、処理対象となるデータを特定する。このようにして、データが特定された後に、MPU36は、CPU12から与えられたインストラクションに応じた処理（たとえば、データの書き込みや読み出し）を実行し、必要な場合には、データをCPU12に伝達する（図9（c）参照）。

【0037】[多空間メモリのより具体的な動作：配列中の要素の削除等]たとえば、ある空間IDをもつ一連のデータ（以下、これを場合によって「配列」と称する。）が、1以上のメモリモジュール14に収容された状態から、特定の要素が削除された状態までの一連の動作につき以下に説明する。あるメモリモジュール14-iにおいて、空間ID「010」に属するデータ群が、図10（a）に示すように格納され、多のメモリモジュール14-jにおいて、空間ID「010」に属するデータ群が、図10（b）に示すように格納されている場合を考える。たとえば、メモリモジュール14-iにおいては、論理アドレス「0」から「59」までのデータが、そのRAMコアの物理アドレス「100」から記憶されていることがわかる。この場合に、みかけの配列は、図10（c）に示すようなものとなる。

【0038】このように複数のメモリモジュールに、ある配列が格納されている場合に、特定の要素を削除する際の処理につき以下に述べる。CPU12から、各メモリモジュール14-1、14-2、…に、制御信号ライン25を介して、空間ID「010」の要素「50～59」を削除するというインストラクションが発せられた場合を考える。図11および図13は、ある空間ID中の所定の範囲の要素を削除するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。

【0039】各メモリモジュールのMPU36は、制御信号ライン25を介して与えられたインストラクションを受理して、その内容を解釈し（ステップ1101）、インストラクション中の「空間ID」を調べ（ステップ1102）、自己のRAMコア34が保持するデータの空間IDに関連しているか否かを判断する（ステップ1103）。ステップ1103にてノー（No）と判断された場合には、処理を終了し、その一方、イエス（Yes）と判断された場合には、MPU36は、空間ID管理テーブルを参照して、当該空間IDに関するデータ群が書き込み可能な状態になっているか、或いは、削除要求のあった範囲のサイズが、全サイズよりも小さいか否かなどを判断する（ステップ1104）。チェックによって異常があると判断された場合（ステップ1105でイエス（Yes））には、MPU36は、制御信号ライン25を介してエラーが生じたことを通知する。その一方、異常がない場合には、MPU36は、インストラクションにより削除を要求された範囲と、自己のRAMコア34に

て保持する要素の範囲とを比較し（ステップ1107）、その比較結果によって（ステップ1108）、種々の処理を実行する。

【0040】まず、削除要求のあった範囲が、自己の保持する要素の範囲よりも後ろである場合（図11の「A」および図12（a）参照）には、MPU36は何ら処理を実行しない（ステップ1109参照）。削除要求のあった範囲が、自己の保持する要素の後方に重なって位置している場合（図11の「B」および図12（b）参照）には、MPU36は、割り付け領域サイズを更新する（ステップ1110）。すなわち、削除要求範囲の先頭（矢印1201参照）から、自己のRAMコア34にて保持する要素の範囲の末尾（矢印1202参照）までがガーページとなるように、割り付け領域サイズが変更される。

【0041】その一方、削除要求のあった範囲が、自己の保持する要素の範囲よりも前方である場合（図11の「C」および図12（c）参照）には、MPU36は、論理開始アドレスを、削除要求のあったサイズ分だけ減じるように、論理開始アドレスを更新する（ステップ1111）。さらに、削除要求のあった範囲が、自己の保持する要素の範囲よりも前方で、かつ、一部だけ重なる場合（図11の「D」および図12（d）参照）には、MPU36は、論理開始アドレスを、削除要求のあった範囲の先頭の値に変更するとともに、物理開始アドレスを、削除要求のあった範囲の末尾の値「+1」に対応する物理アドレスに変更する（ステップ1112）。次いで、MPU36は、割り付け領域サイズを更新する（ステップ1113）。

【0042】また、削除要求のあった範囲が、自己の保持する要素の範囲を包含する場合（図11の「E」および図12（e）参照）には、MPU36は、当該空間IDに関する種々のデータを、空間ID管理テーブルから削除する（図13のステップ1114）。最後に、削除要求のあった範囲が、自己の保持する要素の範囲に包含される場合（図11の「F」および図12（f）参照）には、空間ID管理テーブルを二つに分割して、削除範囲より前方に関する種々のデータと、削除範囲より後方に関する種々のデータに関するものを生成する（ステップ1115）。或いは、MPU36は、自己のRAM34に関して、ガーページコレクションを時刻しても良い。

【0043】このようにして、CPU12からの単一命令（ある空間IDの削除命令）に応答して、各メモリモジュール14が動作して、所定のメモリモジュールにて必要な処理が並列的に実行される。次に、ある空間IDを有する配列の末尾に、ある要素を追加する場合につき簡単に説明する。図14は、ある空間IDの配列の末尾に要素を追加するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。図14のステップ1401～ステップ1



406は、図11のステップ1101～ステップ1106に対応する。次いで、各メモリモジュール14のMPU36は、追加すべき要素を、自己のRAMコア34に記憶すべきか否かを判断する(ステップ1407)。これは、MPU36が、自己の空間ID管理テーブルを参照することにより実現できる。ステップ1407にてイエス(Yes)と判断された場合には、空間ID管理テーブル中の必要な値を更新し(たとえば、割り付け領域サイズを、追加する要素数に応じて変更する)、次いで、RAMセル中の所定の領域に、追加すべき要素を書き込む(ステップ1409)。或いは、空間ID管理テーブルの種々の値を生成して、対応するRAMセル中の領域に、追加すべき要素が書き込まれても良い。

【0044】次いで、MPU36は、空間ID管理テーブル中の当該空間IDに関連する「全サイズ」の値を更新する(ステップ1410)。ステップ1407においてノー(No)と判断された場合にも、空間ID管理テーブル中の関連する「全サイズ」の値が更新される。配列中の任意の位置に要素を追加する場合にも、削除要求と略同等の処理が、各メモリモジュール14にて実行される。

【0045】[多空間メモリのより具体的な動作：配列の結合および分割]次に、図15(a)に示すように、複数の配列を結合したり、或いは、図15(b)に示すように、単一の配列を複数の配列に分割する場合につき説明を加える。本実施の形態にかかるコンピュータシステム10においては、ある空間ID(図15(a)においては空間ID「100」)を有する配列、および/または、他の空間ID(図15(b)においては空間ID「100」)を有する配列が、単一のメモリモジュールのRAMコアに収容されていても良いし、或いは、複数のメモリモジュールのRAMコアに収容されていても良い。図16は、空間ID「10」を有する配列および空間ID「11」を有する配列、並びに、これらがメモリモジュール中に収容された状態を示す図である。図16(a)においては、その空間IDが「10」であり、かつ、各要素のサイズが10ワードである配列1501が示されている。この配列1501中の要素は、メモリモジュール14-1ないし14-xに収容されている。また、図16(b)においては、その空間IDが「11」であり、かつ、各要素のサイズが10ワードである配列1510が示されている。この配列1510の要素も、メモリモジュール14-1ないし14-xに収容されている。

【0046】CPU12が、制御信号ライン25を介して、「空間ID「10」の配列と空間ID「11」の配列とを結合する」旨のインストラクションを発すると、各メモリモジュール14は、これを受理して、自己の保持しているデータの空間IDに関するインストラクションであるか否かを判断する。これらの処理は、図11の

ステップ1101ないしステップ1106と略同様である。次いで、自己の保持しているデータの空間IDが、インストラクションに関連している場合には、メモリモジュールのMPUは、以下の手順にしたがって、配列の結合を実現する。上記図16に示す場合に、関連する各メモリモジュール14は、空間ID「10」および空間ID「11」の双方の要素を保持している場合に、空間ID「11」に関する空間ID管理テーブルの値を更新する。より具体的には、空間ID「10」に関する「全サイズ」の値を参照して、その論理開始アドレスを再度算出する(たとえば、図17の符号1701、1702参照)。また、関連する各メモリモジュールは、空間ID管理テーブル中の「全サイズ」の値を、二つの配列をくみ合わせたサイズに対応するものに更新する(たとえば、図17の符号1703参照)。図17は、このようにして得られた配列1710、および、各メモリモジュール14-1～14-xにおける空間ID管理テーブル(たとえば、符号1711、1712参照)を示す図である。

【0047】図18は、空間ID「10」を有する配列を、空間ID「10」を有する配列と、空間ID「11」を有する配列に分割する一例を示す図である。図18(a)に示す、空間ID「10」を有する配列の分解点を定め、分解点より前方に位置する要素を空間ID「10」の配列とするとともに、分解点より後方に位置する要素を空間ID「11」の配列とする。

【0048】この場合にも、CPU12が、制御信号ライン25を介して、「空間ID「10」の配列を、分解点を境にして、空間ID「10」の配列と空間ID「11」の配列とに分解する」旨のインストラクションを発すると、各メモリモジュール14は、図11のステップ1101ないしステップ1106に略対応する処理を実行し、メモリモジュールのうち、インストラクションに関連するもの(図18の例では、メモリモジュール14-1～14-x)が、所定の処理を実行する。たとえば、MPU36は、分解点より後方に位置する要素を収容している場合に、空間ID「011」に関する空間ID管理テーブル中の種々の値を作成するとともに、空間ID「010」に関する空間管理IDテーブルのうち、全サイズに関する値を更新する。また、分解点より前方に位置する要素のみを収容している場合にも、メモリモジュールのMPU36は、空間ID「010」に関する空間管理IDテーブルのうち、全サイズに関する値を更新する。図19は、このようにして得られた配列1901、1902、および、各メモリモジュール14-1～14-xにおける空間ID管理テーブル(たとえば、符号1911、1912および1913参照)を示す図である。

【0049】[多空間メモリのより具体的な動作：パレルコピー]次に、多空間メモリの下で、場合によって

は組み替え可能バスを利用したパラレルコピーにつき、簡単に説明を加える。たとえば、CPU12からの単一のインストラクションにしたがって、図20に示すように、一方のメモリモジュール群140から、他のメモリモジュール群141へのデータのパラレルコピーを実現することができる。パラレルコピーには以下の態様が考えられる。

【0050】(1)一方のメモリモジュール群140には単一のメモリモジュールが含まれ、他方のメモリモジュール群には、複数のメモリモジュールが含まれる場合。

(2)一方のメモリモジュール群140に、複数のメモリモジュールが含まれ、他方のメモリモジュール群にも、複数のメモリモジュールが含まれる場合。

【0051】前者においては、コピー元の要素を収容しているメモリモジュール14のMPU36は、CPU12から制御信号ライン25を介して与えられたインストラクション(たとえば、ある空間IDを有する配列中の所定の要素を、空間ID8、9、10の配列としてコピーせよという指令)を受理して、RAMコア34から指定された要素を所定のバス上に出力する。その一方、コピー先となるMPU36も、同一のインストラクションの受理に応答して、バスから出力された要素を受理して、これをRAMコア34の所定の領域に記憶するとともに、自己の空間ID管理テーブルを更新する。

【0052】後者においては、複数のバスを利用して、一方のメモリモジュール群140中のメモリモジュールからのデータを、それぞれ、他方のメモリモジュール群141の対応するメモリモジュールに与えることが可能である。この場合には、CPU12は、スイッチ28およびスイッチ30を、所定のメモリモジュール間のデータの授受が可能なように制御すれば良い。

【0053】[多空間メモリのより具体的な動作:隠れ更新など]本実施の形態にかかる多空間メモリを用いて、添字変換により、入力された添え字を変換して、変換済みの添え字によって、配列を指定し、さらに、配列の要素に値を修飾することができる。ある処理が終了して、コミットすることにより、添字変換や値修飾が不要となったときに、各メモリモジュールのMPUは、当該配列に関する空間ID管理テーブルを書きかえてリマッピングを実行することにより、瞬時に添字変換を解消することができる。その一方、値修飾自体は、実際のRAMコアに記憶された要素を更新する必要があるため、時間を要する。したがって、各メモリモジュールにおいて、変換済フラグを設け、値修飾が反映された要素が、実際にRAMコアに記憶した後に、当該要素に対応するフラグを「1」にセットされる。このようにすれば、あるプロセスにおいて、変換フラグを参照して、これが「1」である場合には、値修飾を経る必要がなく、その一方、変換フラグが「0」であるバイには、値修飾を経

る必要があることを容易に知ることができる。したがって、実質的にコミットを瞬時に実現することができる。さらに、本実施の形態にかかる多空間メモリを用いれば、図21に示すように、ネスト構造の値修飾についても、変換済フラグを設け、この変換済フラグを参照することにより、値修飾を経る必要の有無を知ることが可能となる。

【0054】[多空間メモリおよび組み替え可能バスの利用:ソート(その1)]本実施の形態においては、多空間メモリおよび組み替え可能バスを利用することにより、CPU12からの単一のインストラクションに基づき、並列的にソート処理を実行することが可能となる。以下、本実施の形態における並列的なソート処理につき説明を加える。図23および図24は、本実施の形態にかかるソート処理の流れを説明するための図である。このソート処理では、大きく分けて、図23に示す処理(存在数の確定および累計の算出)と、図24に示す処理(レコード番号の転送)とに分けて考えることができる。

【0055】この実施の形態にかかるソート処理を実現するために、本実施の形態においては、レコード番号を格納したレコード番号配列、ある項目に関する実際の項目値を格納した値リスト、および、レコード番号配列からの値(レコード番号)を入力として、対応する値リストの格納位置を示すポインタ値を出力するように構成された値リストへのポインタとを利用している。すなわち、レコード番号から、対応する位置の値リストへのポインタ値が参照され、そのポインタ値にしたがって、実際の項目値が指定されるようになっている(図25参照)。まず、CPU12が、必要なインストラクションを、制御信号ライン25を介して、各メモリモジュール14に与えると、各メモリモジュールにて、図11のステップ1101ないしステップ1106に略同等の処理が実行される。また、関連するメモリモジュールのうち、レコード番号を格納したメモリモジュールからの通知にしたがって、CPU12は、レコード番号を格納した一連のメモリモジュール(第1のメモリモジュール群2301)の出力を、あるバス(「第1のバス」と称する)に接続するように、スイッチ28、30を制御する。

【0056】次いで、値リストへのポインタ配列を格納したメモリモジュールからの通知にしたがって、CPU12は、上記値リストへのポインタ配列を格納した一連のメモリモジュール(第2のメモリモジュール群2302)の出力を、あるバス(「第2のバス」と称する)に接続するように、スイッチ28、30を制御する。さらに、他の一連のメモリモジュール(第3のメモリモジュール群2303)においては、値リストへのポインタと同一サイズ(同じ要素数)の「存在数配列」のための領域が確保され、かつ、各要素が「0」に初期化される。

さらに、第3のメモリモジュール群の入力を、上記第2のバスと接続する。

【0057】次いで、レコード番号配列の先頭から順に、レコード番号が第1のバスに送出される。これは、第1のメモリモジュール群2301において、各メモリモジュールのMPU36が、空間ID管理テーブルを参照して、自己が第1のバスにデータを出力するタイミングを検出して、所定のレコード番号を送出することにより実現される。レコード番号は、第1のバスを介して、第2のメモリモジュール群2302を構成するメモリモジュールの各々に与えられる。各メモリモジュールのMPU36は、自己の空間ID管理テーブルを参照して、自己が管理する値リストへのポインタ配列に関連するレコード番号が入力されたことを検出し、当該入力に対応するポインタ値を第2のバスに出力する。

【0058】ポインタ値は、第2のバスを介して、第3のメモリのジュール群を構成するメモリモジュールの各々に与えられる。各メモリモジュールのMPU36は、自己の空間ID管理テーブルを参照して、自己が管理する値リストのポインタ配列に関連するポインタ値が与えられたことを検出し、存在数配列において、ポインタ値に対応する位置の要素をインクリメントする。この動作を繰り返すことにより、項目値が何度レコード番号により指されているか（ポイントされているか）を知ることができる。上記存在数配列のための一連の処理が終了すると、ソートされたレコード番号を格納する配列を作成するために、一連のメモリモジュールに、一定の領域が確保される。この一連のメモリモジュールを、第4のメモリモジュール群2304と称する。CPU12は、先の処理に利用した第3のメモリモジュール群の出力と、第4のメモリモジュール群の入力とを、バス（「第3のバス」と称する）を介して接続するように、スイッチ28、30を制御する。

【0059】このような準備が終了した後に、ソート処理が実行される。より具体的には、レコード番号配列の先頭から、レコード番号が第1のバスを介して、第2のメモリモジュール群を構成するメモリモジュールに与えられる。第2のメモリモジュール群中の所定のメモリモジュールにおいては、MPU36がレコード番号の受理に応答して、ポインタ値を、第2のバスを介して、第3のメモリモジュール群に伝達する。次いで、第3のメモリモジュール群のうち、所定のメモリモジュールにおいて、MPU36が、ポインタ値に基づき、関連する存在数配列を参照して、レコード番号の格納位置を決定する。これにより、レコード番号およびその格納位置が、当該メモリモジュールから、第3のバスに送出される。したがって、第4のメモリモジュール群の所定のメモリモジュールにおいて、MPU36が、レコード番号を、所定の格納位置に配置する。この処理を繰り返すことにより、第4のメモリモジュール群に、ソートされたレコード番

号の配列（図24の符号2410）を作成することができる。

【0060】たとえば、図23に示す処理を、パイプライン処理にすることができる。すなわち、第1のバスにおいて、あるレコード番号「p」が伝達されている際に、第2のバスにおいては、レコード番号「p-1」に関するポインタ値「P(p-1)」が伝達され得る。また、同様に、図24に示す処理も、パイプライン処理にすることが可能である。この場合にも、第1のバスにおいて、あるレコード番号「p」が伝達されている際に、第2のバスにおいては、レコード番号「p-1」に関するポインタ値「P(p-1)」が伝達され得る。さらに、同じタイミングで、第3のバスにおいては、レコード番号「p-1」に関する格納位置が伝達され得る。

【0061】このようなパイプライン処理の処理時間につき、以下のような結果が得られた。まず、図23の処理に関して、第1のバスないし第4のバスが、それぞれ、128ビットであり、それぞれ、12.8GB/秒の転送能力があると考え、また、レコード番号やポインタ値が、それぞれ、32ビット整数であると仮定した。いま、レコード数が10億個の場合に、上記処理では、40億バイトの転送が発生するが、パイプライン処理を実行するため、 $4G/12.8G=0.3125$ 秒にて完了することが分かった。同様に、図24の処理に関して、同様の転送能力およびデータサイズを仮定すると、レコード数が10億個の場合に、80億バイトの転送が発生するが、本実施の形態によれば、パイプライン処理の実行により、 $8G/12.8G=0.625$ 秒にて処理を完了することができる。

【0062】〔多空間メモリおよび組み替え可能バスの利用：ソート（その2）〕次に、他の手法によるソート処理につき簡単に説明を加える。このソート処理においても、まず、レコード番号配列を格納したメモリモジュールからなる第1のメモリモジュール群（図26の符号2601参照）の出力と、第1のバスとが接続され、かつ、値リストへのポインタ配列を格納したメモリモジュールからなる第2のメモリモジュール群2602の入力が、第1のバスと接続される。これにより、第1のメモリモジュール群2601の出力が、第1のバスを介して、第2のメモリモジュール群2602に伝達可能となる。その一方、第2のメモリモジュール群2602と、同一の数の空間IDを有する配列の領域が、第3のメモリモジュール群2603に確保されるとともに、第2のメモリモジュール群2602の出力と、第3のメモリモジュール群の入力とが、第2のバスを介して接続される。

【0063】次いで、第1のメモリモジュール群2601において、あるレコード番号を収容するメモリモジュールのMPU36が、当該レコード番号を、第1のバスに送出すると、第2のメモリモジュール群2602の所定

のメモリモジュールにおいて、MPU 36がこの受理に  
応答して、対応するポインタ値から、空間IDを算出  
し、レコード番号および空間IDを、第2のバスに送出  
する。

【0064】第3のメモリモジュール群において、当該  
空間IDおよびレコード番号に基づき、所定のメモリモ  
ジュール36が起動し、当該空間IDを有する配列の末  
尾に、与えられたレコード番号を配置する。このような  
処理を全てのレコード番号について実行した後に、第3  
のメモリモジュール群において、各メモリモジュールの  
MPU 36は、自己の有する配列を結合するための処理  
を実行する。このような手法によっても、高速なソート  
処理を実現することができる。

【0065】〔多空間メモリおよび組み替え可能バスの  
利用：検索（その1）〕また、本実施の形態において  
は、多空間メモリおよび組み替え可能バスを利用するこ  
とにより、CPU 12からの単一のインストラクション  
に基づき、並列的に検索処理を実行することができる。  
図27および図28は、本実施の形態にかかる検索処理  
の流れを説明するための図である。この検索処理のため  
に、レコード番号配列、値リストへのポインタ配列、値  
リストおよび後述する可否フラグ配列などが利用され  
る。したがって、この例でも、図25のように、レコー  
ド番号、ポインタ値、項目値の順に、値が参照されるよ  
うになっている。

【0066】まず、CPU 12が、必要なインストラク  
ションを、制御信号ライン25を介して、各メモリモジ  
ュール14に与えると、各メモリモジュールにて、図1  
1のステップ1101ないしステップ1106に略同等  
の処理が実行される。また、関連するメモリモジュール  
のうち、値リストを格納したメモリモジュールからの通  
知にしたがって、CPU 12は、値リストを格納した一  
連のメモリモジュール（第1のメモリモジュール群27  
01）の出力を、あるバス（「第1のバス」と称する）  
に接続するように、スイッチ28、30を制御する。さら  
に、そのよう素数が値リストのものと同一である可否  
フラグ配列のための領域が、一連のメモリモジュール  
（第2のメモリモジュール群2702）に確保され、当  
該第2のメモリモジュール2702に属する各メモリモ  
ジュールのMPU 36が、当該領域の要素を「0」に初  
期化する。

【0067】次いで、第2のメモリモジュール群270  
2の入力が、第1のバスに接続される。次いで、CPU  
12から与えられた検索条件にしたがって、第2のメモ  
リモジュール群の各メモリモジュールにおいて、MPU  
36が、値リスト中の検索条件に合致する項目値の位置  
を参照して、可否フラグ配列の対応する値を「1」にセ  
ットする。たとえば、検索条件が範囲であれば、二分割  
法などを用いれば良い。また、その他の条件であれば、  
要素ごとにその可否を判断すれば良い。このような処理

が終了した後に、検索が実行される。まず、レコード番  
号配列を格納した一連のメモリモジュール（第3のメモ  
リモジュール群2703）の出力を、第1のバスに接続  
するとともに、値リストへのポインタ配列を格納した一  
連のメモリモジュール（第4のメモリモジュール群27  
04）の入力を、第1のバスに接続するよう、CPU 1  
2は、スイッチ28、30を制御する。また、第4のメ  
モリモジュール群2704の出力を、第2のメモリモジ  
ュール群2702の入力とを、第2のバスと接続するよ  
うに、CPU 12は、スイッチ28、30を制御する。

【0068】さらに、レコード番号の要素数と同じ要素  
数を有する配列のための領域が、一連のメモリモジ  
ュール（第5のメモリモジュール2705）に確保され、C  
PU 12は、その入力と、第2のメモリモジュール群2  
702の出力とが、第3のバスを介して接続されるよう  
に、スイッチ28、30を制御する。このような処理の  
後に、レコード番号配列の先頭から順に、レコード番号  
が第1のバスに送出される。これは、第3のメモリモジ  
ュール群2703において、各メモリモジュールのMP  
U 36が、空間ID管理テーブルを参照して、自己が第  
1のバスにデータを出力するタイミングを検出して、所  
定のレコード番号を送出することにより実現される。

【0069】レコード番号は、第1のバスを介して、第  
4のメモリモジュール群2704を構成するメモリモジ  
ュールの各々に与えられる。各メモリモジュールのMP  
U 36は、自己の空間ID管理テーブルを参照して、自  
己が管理する値リストへのポインタ配列に関連するレコ  
ード番号が入力されたことを検出し、受理したレコー  
ド番号および当該入力に対応するポインタ値を第2のバ  
スに出力する。ポインタ値は、レコード番号とともに、第  
2のバスを介して、第3のメモリーのジュール群を構成  
するメモリモジュールの各々に与えられる。各メモリモ  
ジュールのMPU 36は、自己の空間ID管理テーブル  
を参照して、自己が管理する可否フラグ配列の位置と同  
じ位置を示すポインタ値が与えられたことを検出し、当  
該ポインタ値が示す可否フラグが、「0」であるか  
「1」であるかを判断する。次いで、可否フラグが  
「1」の場合には、関連するレコード番号が、第3のバ  
スを介して、第5のメモリモジュール群2705に与え  
られる。

【0070】第5のメモリモジュール群2705におい  
ては、各メモリモジュールのMPU 36は、自己の空間  
ID管理テーブルを参照して、自己が管理するヒット情  
報格納用配列の位置と同じ位置を示すレコード番号が与  
えられたことを検出し、その位置の要素を「1」にす  
る。このような処理を所定のレコード番号に関して繰り  
返し、ヒット情報格納用配列にて「1」である要素を取  
り出すことにより、検索が完了する。

【0071】ソート処理と同様に、上記検索処理でも、  
図27を参照して説明した処理、および、図28を参照

して説明した処理を、それぞれ、パイプライン処理にて実現することができる。検索処理におけるパイプライン処理の処理時間につき、以下のような結果が得られた。バスの転送能力、および、各要素のビット数は、ソート処理と同様であると考えた。レコード数が10億個の場合に、上記検索処理では、80億バイトの転送が発生するが、パイプライン処理を実行するため、 $8G/12$ 、 $8G=0.624$ 秒にて完了することが分かった。

【0072】さらに、本検索処理を用いれば、AND、OR或いはNOTなどを組み合わせた複数項目の検索を実現することもできる。より具体的には、各項目につき、ヒット情報格納用配列を作成し、これら配列の要素間での論理演算を行えば良い。たとえば、二つの項目のAND或いはOR検索では、ヒット情報格納用配列の要素の転送(10億バイト)が行われる。したがって、その処理時間は、 $(10G/8)/12$ 、 $8G=0.098$ 秒だけ必要であることが理解できる。

【0073】なお、さらに高速化を図るために、AND検索の場合には、二つの検索処理を実行するメモリモジュール群を縦列に接続すれば良い。また、第4のメモリモジュール群と第2のメモリモジュール群を、同一の複数のメモリモジュールにて構成できるように、配列を配置すれば、ボトルネックを解消することができ、これにより、略2倍の処理速度を得ることが可能となる。

【0074】本発明は、以上の実施の形態に限定されることなく、特許請求の範囲に記載された発明の範囲内で、種々の変更が可能であり、それらも本発明の範囲内に包含されるものであることは言うまでもない。たとえば、前記実施の形態においては、本発明を、コンピュータシステムに適用しているがこれに限定されるものではなく、パーソナルコンピュータなどに接続可能なコンピュータボードに適用することもできる。この場合には、図1において、CPU12、メモリユニット14、バス24等がボード上に搭載され、これが、本発明における情報処理ユニットを構成する。

【0075】また、CPU12とメモリモジュール14との間、および/または、メモリモジュール14間を接続するバスの組の数は、前記実施の形態に限定されるものではなく、コンピュータシステムを搭載する回路基板の大きさ、各バスのビット数などを考慮して適宜決定することができる。また、前記実施の形態においては、メモリモジュールの入出力とバスとの接続を規定するためのスイッチ28と、CPUとメモリモジュールとの間、メモリモジュール間で、バスの切断することができるスイッチ30とを設けている。スイッチ30を設けることにより、たとえば、あるバス(図1のバス24-4参照)を、CPUモジュール12とメモリモジュール14-1とのデータ授受のために利用するとともに、同時に、メモリモジュール14-2とメモリモジュール14-3との間のデータ授受のために利用することができる

(この場合に、スイッチ30-5をオフにすれば良い)。したがって、より有効にバスを利用することが可能となっている。しかしながら、バスの組を数を十分に大きくできる場合、或いは、メモリモジュールの数が比較的少ない場合には、スイッチ30を必ずしも設けなくて良い。

【0076】また、本明細書において、制御信号ライン25を介して、CPU12からのインストラクションが与えられることを記載したが、制御信号ライン25を介して、インストラクションのほか、クロックなど、各メモリモジュールが同期して作動するための種々の制御信号が与えられ、かつ、各メモリモジュールからCPU12への所定の信号(たとえば、エラー信号や、データ受理を示す信号)が与えられていることは言うまでもない。さらに、本明細書において、一つの手段の機能が、二つ以上の物理的手段により実現されても、若しくは、二つ以上の手段の機能が、一つの物理的手段により実現されてもよい。

【0077】

【発明の効果】本発明によれば、分散メモリー型において、単一命令により種々のメモリーに記憶された配列中の要素を入出力し、著しく高速な並列処理を実現可能なコンピュータアーキテクチャを提供することが可能となる。

【図面の簡単な説明】

【図1】 図1は、本発明の実施の形態にかかるコンピュータシステムの構成を示すブロックダイヤグラムである。

【図2】 図2は、本実施の形態にかかるメモリモジュールの概略を示すブロックダイヤグラムである。

【図3】 図3は、単一メモリ空間における一連のデータの配置を示す図である。

【図4】 図4は、本発明に係る多空間メモリにおける一連のデータの配置を示す図である。

【図5】 図5は、本実施の形態におけるアドレスリマッピングを説明するための図である。

【図6】 図6は、本実施の形態における値修飾を説明するための図である。

【図7】 図7は、本実施の形態にかかるメモリモジュール間のパイプライン処理の概略を示す図である。

【図8】 図8は、本実施の形態にかかる多空間メモリの下での、メモリモジュール14の構造を説明するための図である。

【図9】 図9は、多空間メモリの下での、メモリモジュール14の構造を説明するための図である。

【図10】 図10は、多空間メモリの下での、メモリモジュール14の構造を説明するための図である。

【図11】 図11は、ある空間ID中の所定の範囲の要素を削除するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャ

ートである。

【図 12】 図 12 は、削除される要素と、メモリモジュールにて保持している要素の配置との関係を示す図である。

【図 13】 図 13 は、ある空間 ID 中の所定の範囲の要素を削除するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。

【図 14】 図 14 は、ある空間 ID の配列の末尾に要素を追加するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。

【図 15】 図 15 は、本実施の形態にかかる配列の結合および配列の分割を説明するための図である。

【図 16】 図 16 は、本実施の形態において、空間 ID「10」を有する配列および空間 ID「11」を有する配列、並びに、これらがメモリモジュール中に収容された状態を示す図である

【図 17】 図 17 は、本実施の形態において、配列の結合により得られた配列、および、各メモリモジュールにおける空間 ID 管理テーブルを示す図である。

【図 18】 図 18 は、本実施の形態において、空間 ID「10」を有する配列を、空間 ID「10」を有する配列と、空間 ID「11」を有する配列に分割する一例を示す図である。

【図 19】 図 19 は、本実施の形態において、配列の分割により得られた配列、および、各メモリモジュールにおける空間 ID 管理テーブルを示す図である。

【図 20】 図 20 は、本実施の形態にかかる、一方のメモリモジュール群から、他のメモリモジュール群へのデータの平行コピーを示す図である。

【図 21】 図 21 は、本実施の形態にかかる変換済み\*

\* フラグの利用を説明するための図である。

【図 22】 図 22 は、本実施の形態にかかる変換済みフラグの利用を説明するための図である。

【図 23】 図 23 は、本実施の形態にかかるソート処理の流れを説明するための図である。

【図 24】 図 24 は、本実施の形態にかかるソート処理の流れを説明するための図である。

【図 25】 図 25 は、本実施の形態において、レコード番号から項目値が特定されるまでのデータの参照手順を示す図である。

【図 26】 図 26 は、本実施の形態にかかる他のソート処理の流れを説明するための図である。

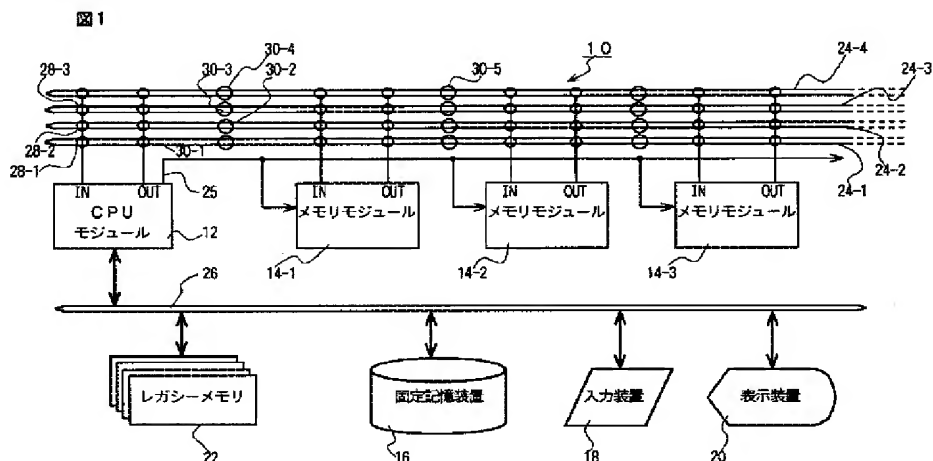
【図 27】 図 27 は、本実施の形態にかかる検索処理の流れを説明するための図である。

【図 28】 図 28 は、本実施の形態にかかる検索処理の流れを説明するための図である。

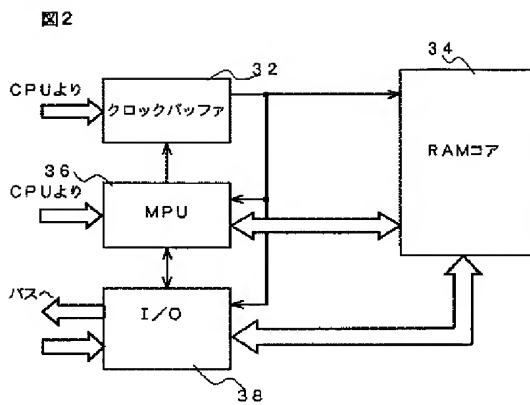
【符号の説明】

10	コンピュータシステム
12	CPUモジュール
14	メモリモジュール
16	固定記憶装置
18	入力装置
20	表示装置
22	レガシーメモリ
24	バス
25	制御信号ライン
26	バス
28、30	スイッチ
32	クロックバッファ
34	RAMコア
36	MPU
38	I/O

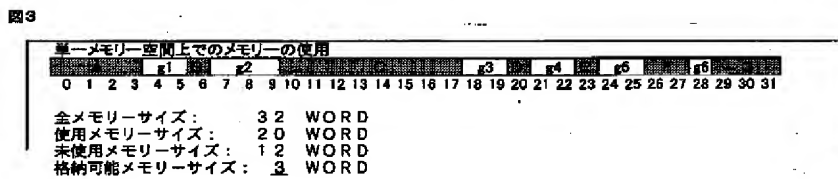
【図 1】



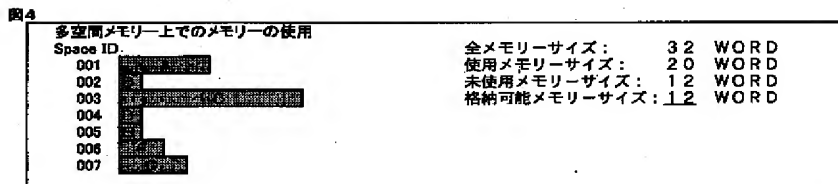
【図2】



【図3】

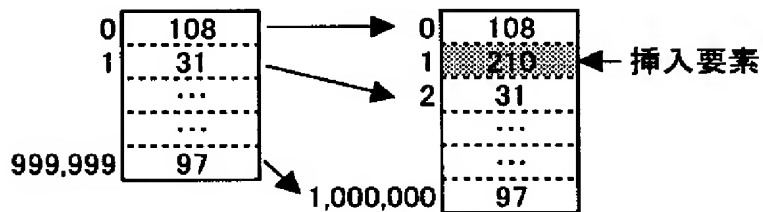


【図4】

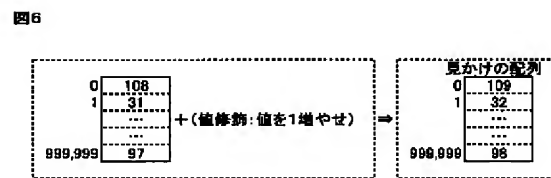


【図5】

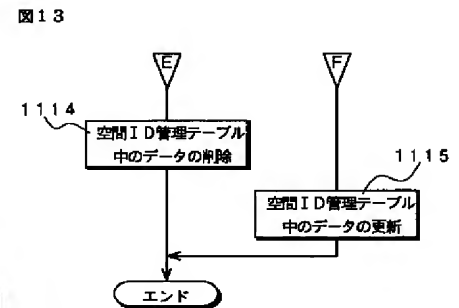
図5



【図6】



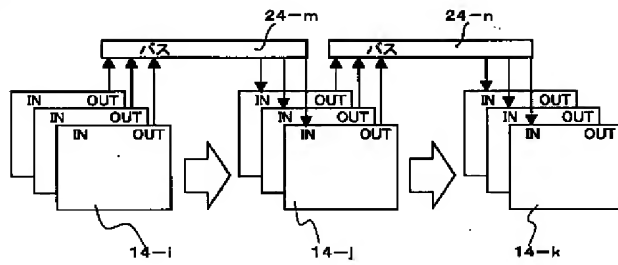
【図13】





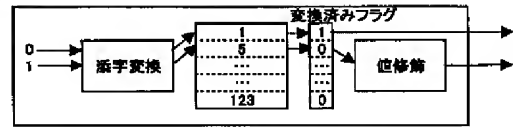
【図7】

図7



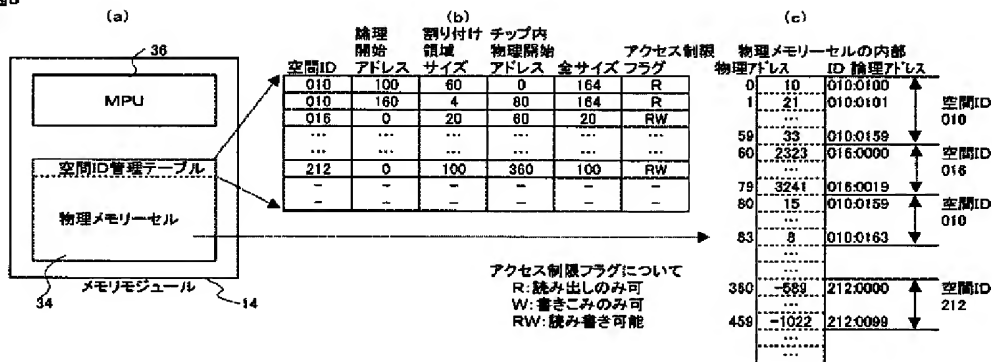
【図21】

図21



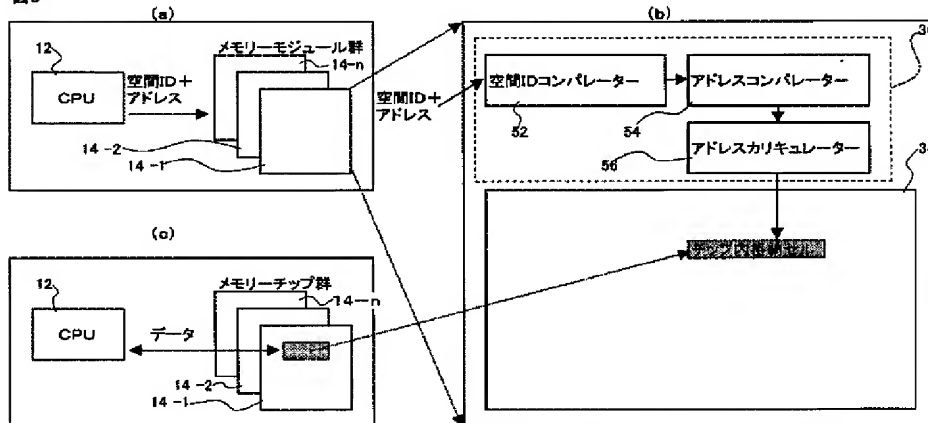
【図8】

図8



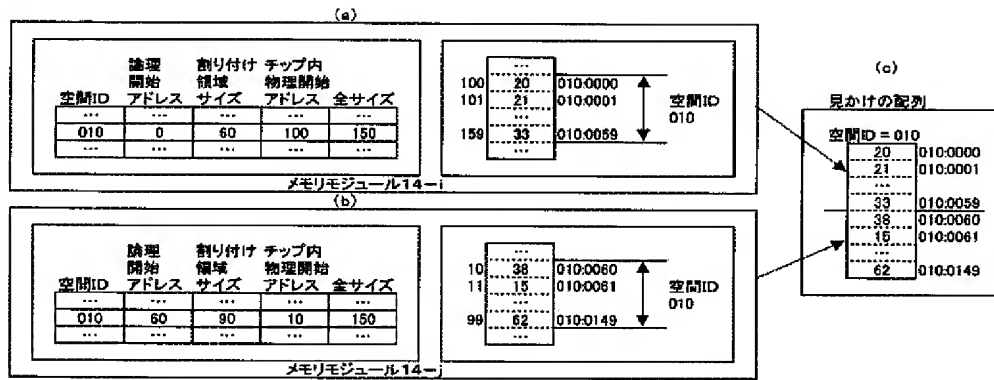
【図9】

図9



【図10】

図10



【図12】

【図15】

図12

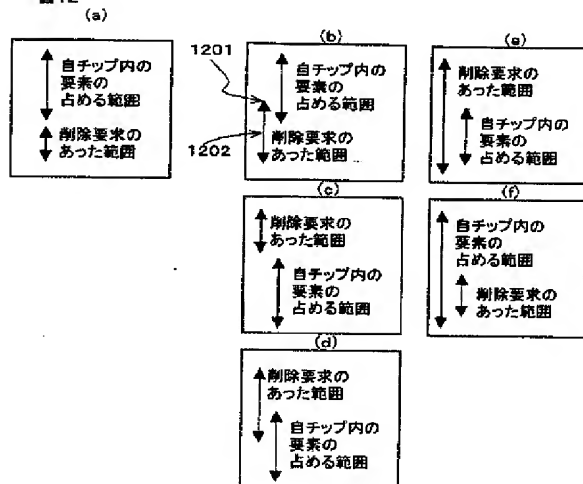
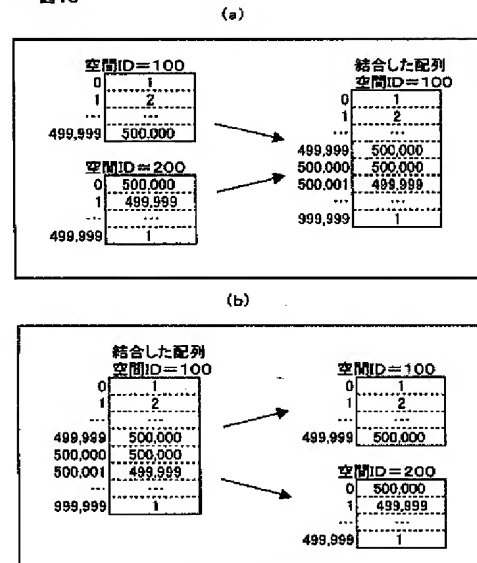


図15



【図17】

【図25】

図17

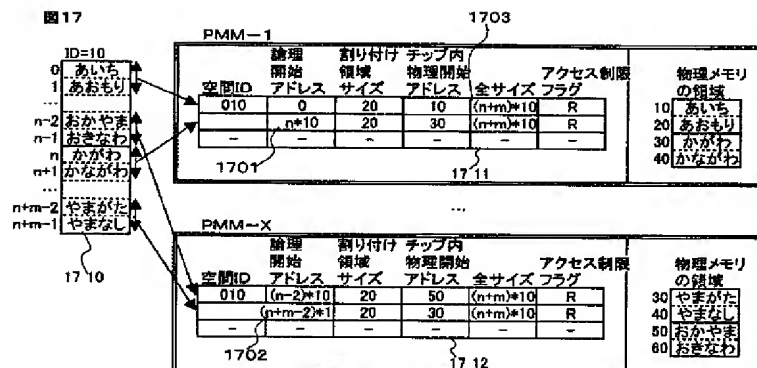
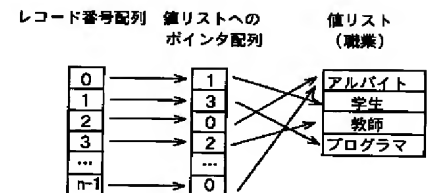


図25



【図11】

図11

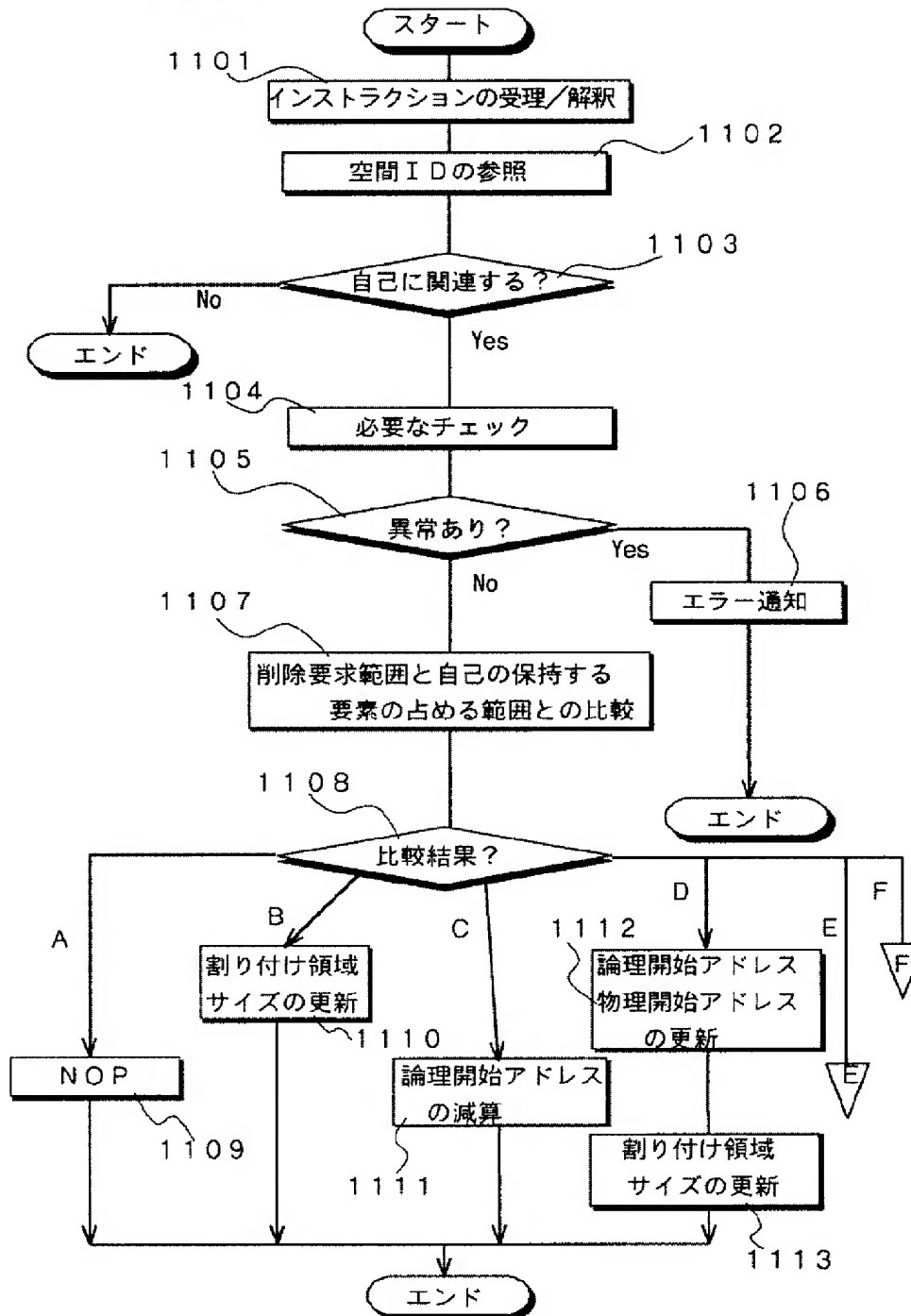
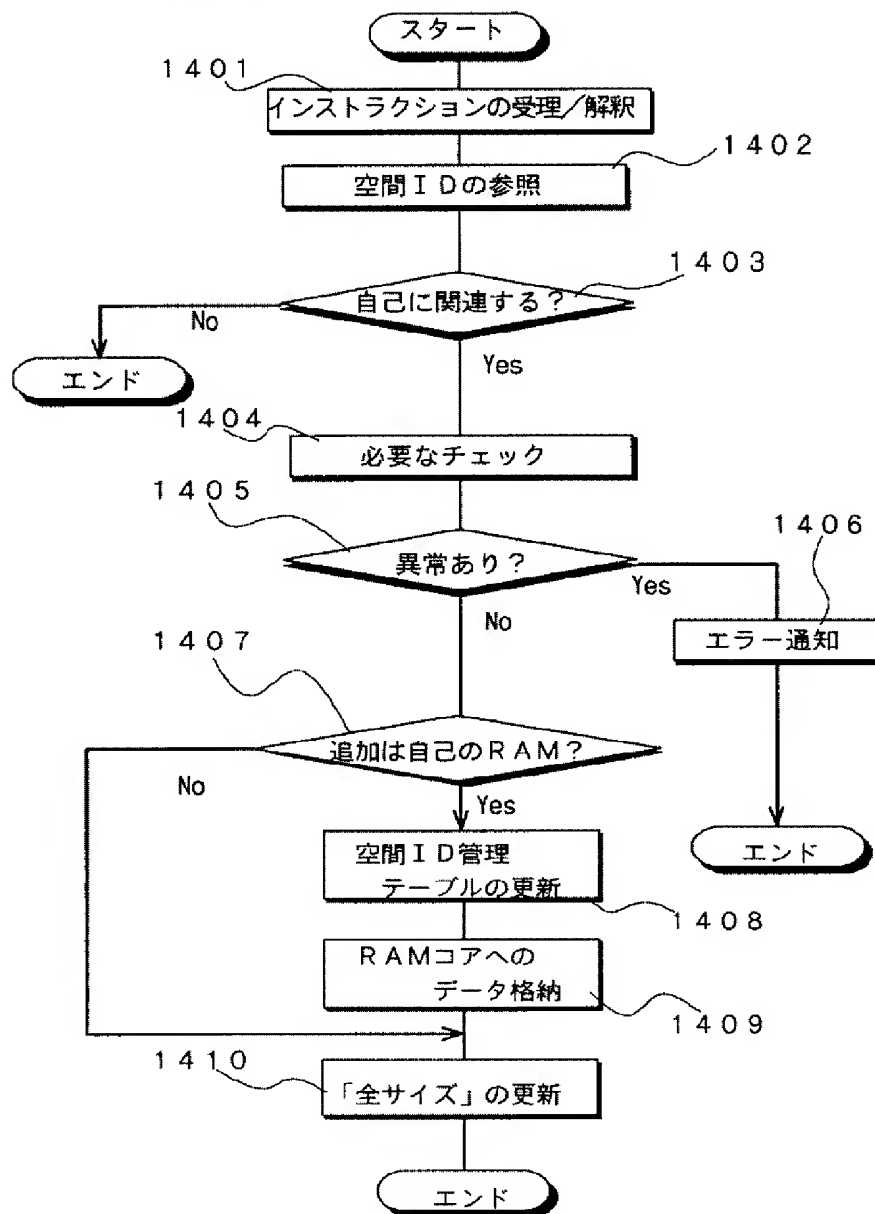
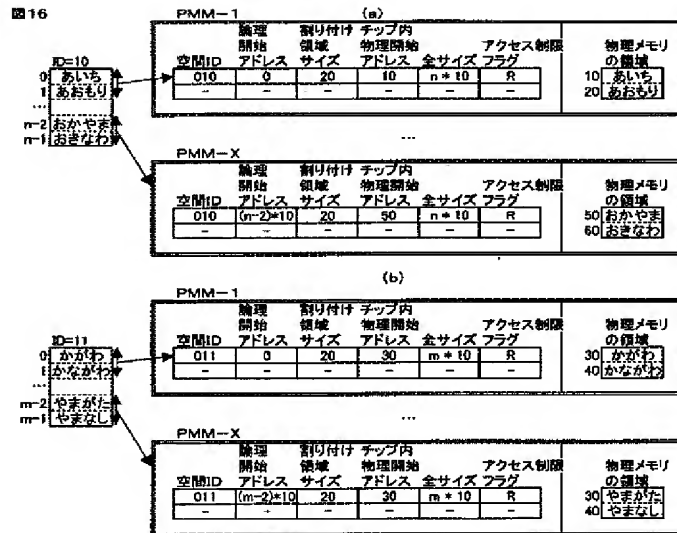


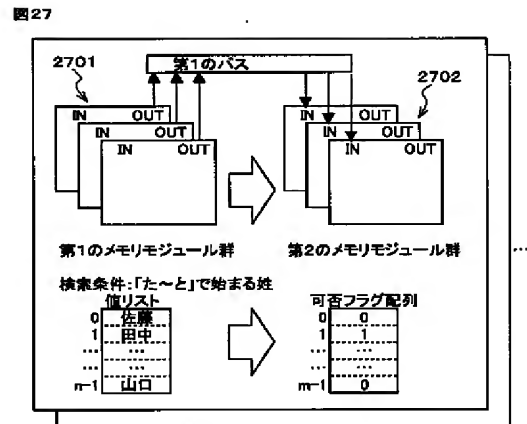
图 14



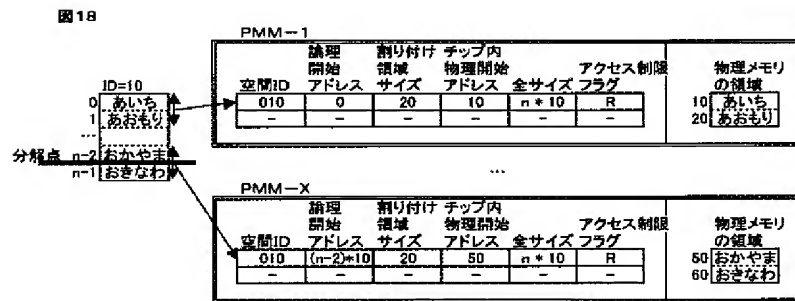
【図16】



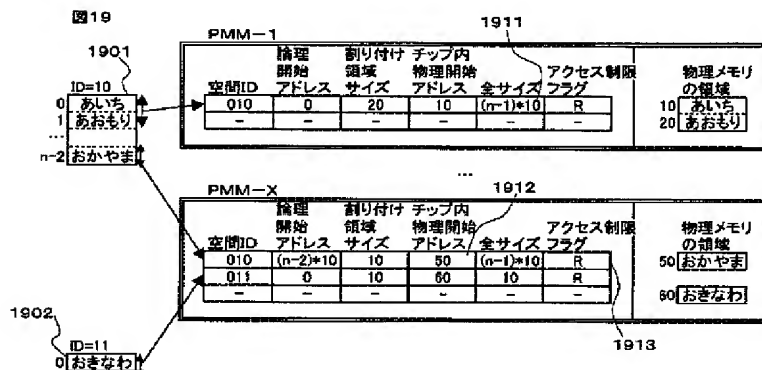
【図27】



【図18】

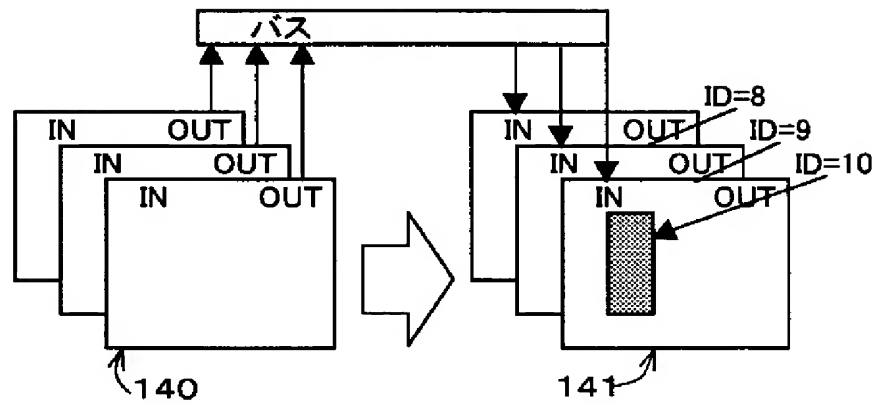


【図19】



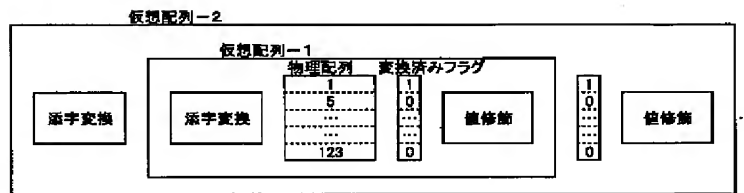
【図20】

図20



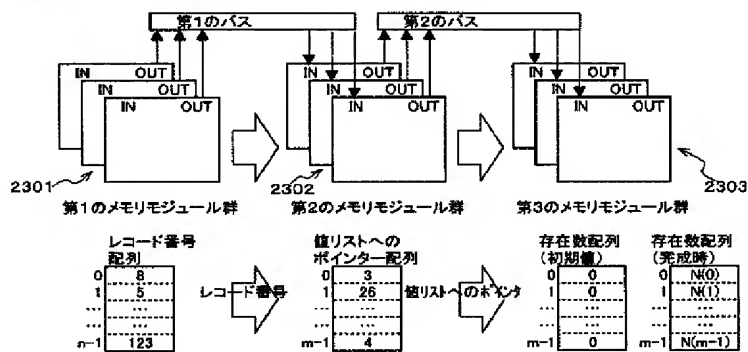
【図22】

図22



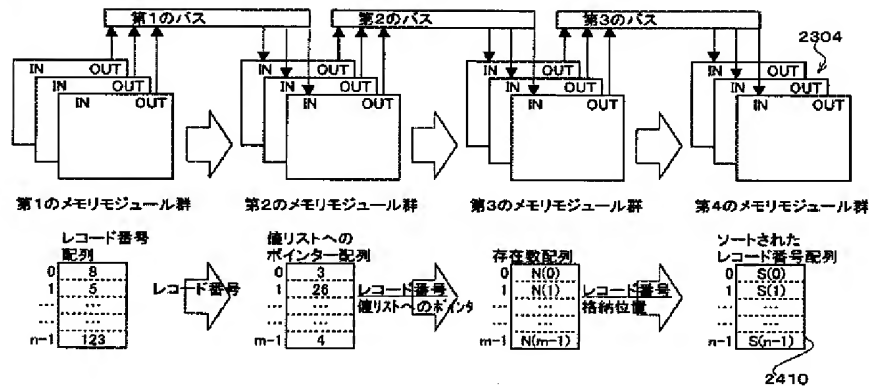
【図23】

図23



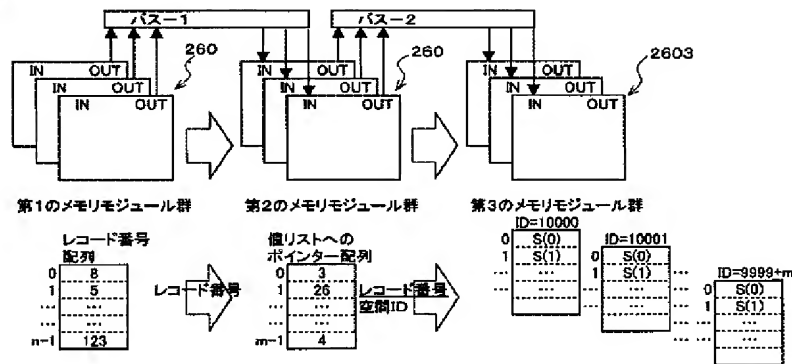
【図24】

図24



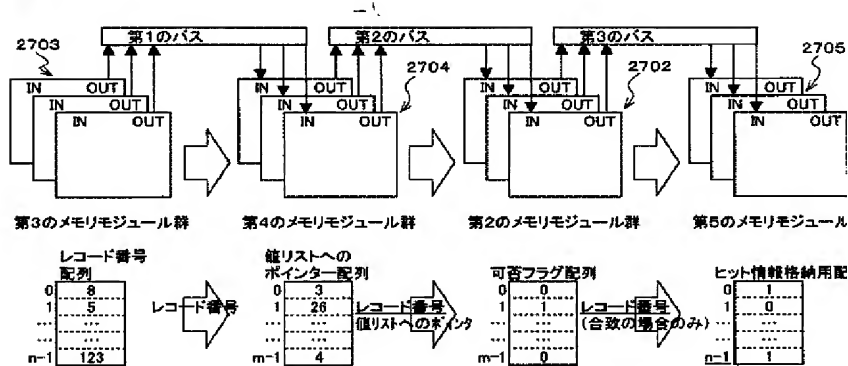
【図26】

図26



【図28】

図28





フロントページの続き

(51)Int.Cl. <sup>7</sup>		識別記号	F I	テーマコード (参考)	
G 0 6 F	13/16	5 1 0	G 0 6 F	13/16	5 1 0 D
	15/16	6 1 0		15/16	6 1 0 A